

Exhibit 24 to Amended Complaint
Intellectual Ventures I LLC and Intellectual Ventures II LLC

Example Southwest Count XI Systems and Services
U.S. Patent No. 7,721,282 (“the ’282 Patent”)

The Accused Systems and Services include without limitation Southwest systems and services that utilize Docker; all past, current, and future systems and services that operate in the same or substantially similar manner as the specifically identified systems and services; and all past, current, and future Southwest systems and services that have the same or substantially similar features as the specifically identified systems and services (“Example Southwest Count XI Systems and Services” or “Southwest Systems and Services”).¹

On information and belief, the Southwest Systems and Services use Docker in Southwest’s private cloud(s). For example, Southwest posts, or has posted, job opportunities that require familiarity with Docker containerization concepts.

See <https://www.linkedin.com/in/charlesmarshall-eth/>, the job profile of an associate software engineer listing Docker as a skill. (Last accessed on 9/23/2024).

See <https://www.linkedin.com/in/bhaveskhar-kongari/>, the job profile of a senior DevOps Engineer listing usage of Docker. (Last accessed on 9/23/2024).

As another example, Southwest has stated that it is investing in cloud technology and has “moved about 50% of its technology” to the cloud and has indicated cloud migration is one of its areas of focus for 2024 and beyond.

Source: <https://www.phocuswire.com/southwest-airlines-cio-tech-investment>.

¹ Plaintiffs do not accuse the public clouds of Defendant, to the extent those services are provided by a cloud provider with a license to Plaintiffs’ patents that covers Defendant’s activities. [REDACTED]

[REDACTED] Plaintiffs do not accuse the public clouds of Defendants if those services are provided by a cloud provider with a license to Plaintiffs’ patents that covers Defendants’ activities. [REDACTED]

[REDACTED] Plaintiffs accuse Defendant private clouds that implement Docker and non-licensed public clouds that Defendant uses to support Docker for its systems and services. [REDACTED]



Top Airlines, Airports & Air Services Companies Using Docker

43,233 companies using this technology

By [Docker](#)

Docker is a software container platform. Developers use Docker to eliminate “works on my machine” problems when collaborating on code with co-workers. Operators use Docker to run and manage apps side-by-side in isolated containers to get better compute density. Enterprises use Docker to build agile software delivery pipelines. [Read less](#)

15 ↑

Companies adopted this tech
in the last month

189 ↓

Companies dropped this tech
in the last month

33.99% ⓘ

The market share of the
technology in its category

4



Southwest Airlines

Employee count: 72,450



Southwest Airlines Co. operates as a passenger airline company that provides scheduled air transportation services in the United States and near-international markets. As of December 31, 2023, the company operated a total fleet of 817 Boeing 737 airc ... [Read more](#)

Source: <https://www.zoominfo.com/tech/23717/docker-tech-from-transportation-airline-industry-in-us-by-revenue>.²

² Unless otherwise noted, all sources cited in this document were publicly accessible as of the filing date of the Complaint.

U.S. Patent No. 7,721,282 (Claim 1)	
Claim	Example Southwest Count XI Systems and Services
1. A system for distributing an application environment comprising:	<p>To the extent this preamble is limiting, on information and belief, the Southwest Count XI Systems and Services are a “system for distributing an application environment.”</p> <p>Docker is an open-source platform that enables the development and distribution of applications and facilitates running applications in an isolated environment called containers. A multi-server environment capable of deploying and running Docker containers is considered a system.</p> <div data-bbox="688 592 1841 930"> <h2>Docker overview</h2> <p>Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications. By taking advantage of Docker's methodologies for shipping, testing, and deploying code, you can significantly reduce the delay between writing code and running it in production.</p> </div> <p>Source: https://docs.docker.com/guides/docker-overview/.³ (Last accessed on May 15, 2025.)</p>

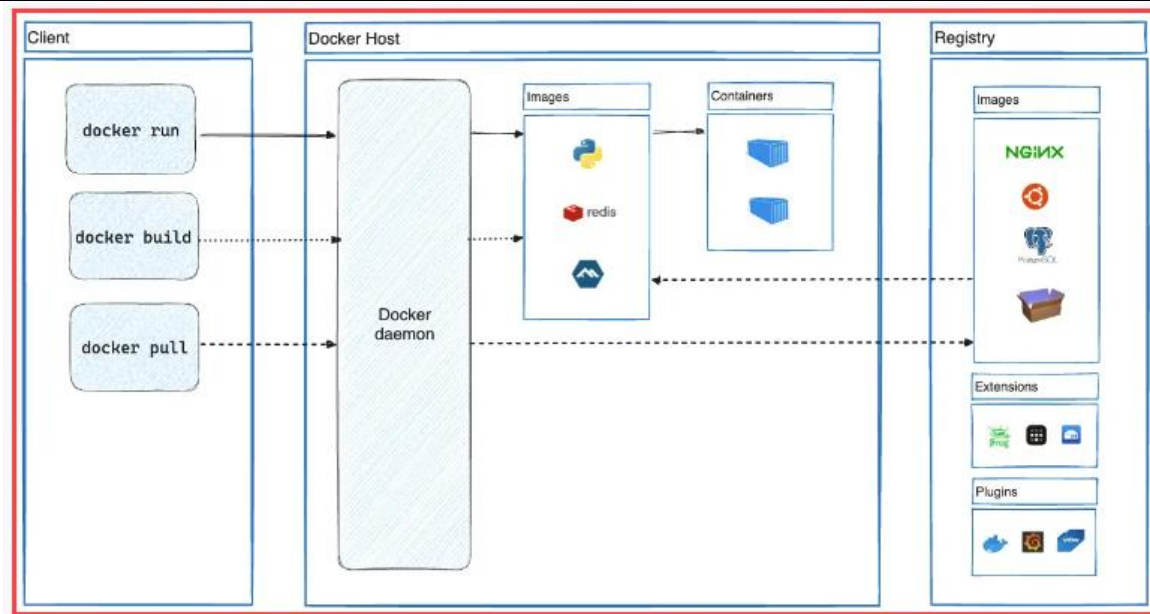
³ Annotations added unless otherwise noted.

U.S. Patent No. 7,721,282 (Claim 1)	
Claim	Example Southwest Count XI Systems and Services
	<div><h2>The Docker platform</h2><p>Docker provides the ability to package and run an application in a loosely isolated environment called a container. The isolation and security lets you run many containers simultaneously on a given host.</p><p>Containers are lightweight and contain everything needed to run the application, so you don't need to rely on what's installed on the host. You can share containers while you work, and be sure that everyone you share with gets the same container that works in the same way.</p><p>Docker provides tooling and a platform to manage the lifecycle of your containers:</p><ul style="list-style-type: none">• Develop your application and its supporting components using containers.• The container becomes the unit for distributing and testing your application.• When you're ready, deploy your application into your production environment, as a container or an orchestrated service. This works the same whether your production environment is a local data center, a cloud provider, or a hybrid of the two.<p>Source: https://docs.docker.com/guides/docker-overview/. (Last accessed on May 15, 2025).</p><p>For example, a user can interact with Docker on the command line to run, build, pull, and/or configure containers, images, and/or volumes.</p></div>

U.S. Patent No. 7,721,282 (Claim 1)

Claim

Example Southwest Count XI Systems and Services

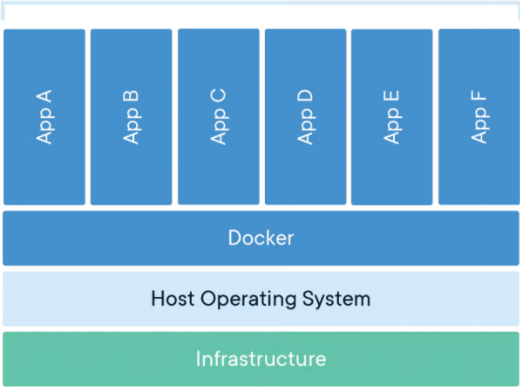


Source: <https://docs.docker.com/guides/docker-overview/>. (Last accessed on May 15, 2025).

1[a] a compute node comprising a computer system;

On information and belief, the Southwest Count XI Systems and Services include “a compute node comprising a computer system.”

Docker requires a host environment or a server to host containers in isolation from each other. Containers (e.g., a compute node) are running instances of an image, having executable packages of software, including code, runtime, system tools, libraries, and settings (e.g., comprising a computer system).

U.S. Patent No. 7,721,282 (Claim 1)	
Claim	Example Southwest Count XI Systems and Services
	<p>A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another. A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.</p> <p>Source: https://www.docker.com/resources/what-container/. (Last accessed on May 15, 2025).</p> <div><p>Containerized Applications</p><p>The diagram illustrates the architecture of containerized applications. It consists of four stacked layers. The top layer, labeled 'Containerized Applications', contains six blue rectangular boxes, each representing an application: 'App A', 'App B', 'App C', 'App D', 'App E', and 'App F'. Below these applications is a dark blue layer labeled 'Docker'. Underneath the Docker layer is a light blue layer labeled 'Host Operating System'. The bottom layer is a green layer labeled 'Infrastructure'.</p></div> <p>Source: https://www.docker.com/resources/what-container/. (Last accessed on May 15, 2025).</p>

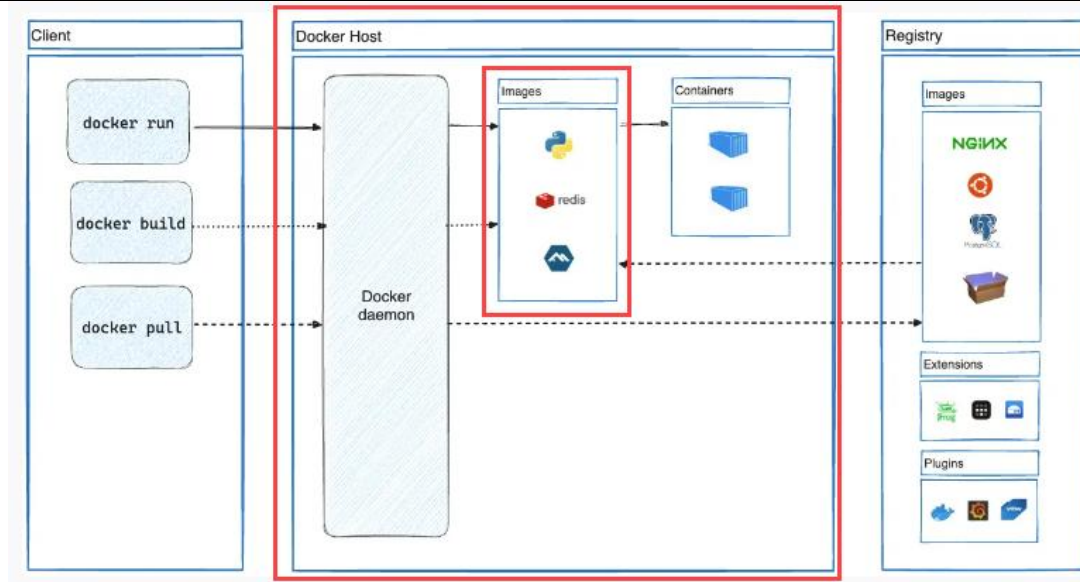
U.S. Patent No. 7,721,282 (Claim 1)	
Claim	Example Southwest Count XI Systems and Services
	<p>The Docker platform</p> <p>Docker provides the ability to package and run an application in a loosely isolated environment called a container. The isolation and security lets you run many containers simultaneously on a given host. Containers are lightweight and contain everything needed to run the application, so you don't need to rely on what's installed on the host. You can share containers while you work, and be sure that everyone you share with gets the same container that works in the same way.</p> <p>Source: https://docs.docker.com/get-started/docker-overview/. (Last accessed on May 15, 2025).</p> <p>By default, a container is relatively well isolated from other containers and its host machine. You can control how isolated a container's network, storage, or other underlying subsystems are from other containers or from the host machine.</p> <p>Source: https://docs.docker.com/get-started/docker-overview/. (Last accessed on May 15, 2025).</p> <p>For example, containers are running instances of images. Docker images include the libraries, dependencies, and other environment elements for an application to run in. The storage driver controls how the images and container instances are resident on the host.</p> <p>Images and containers</p> <p>Fundamentally, a container is nothing but a running process, with some added encapsulation features applied to it in order to keep it isolated from the host and from other containers. One of the most important aspects of container isolation is that each container interacts with its own private filesystem; this filesystem is provided by a Docker image. An image includes everything needed to run an application - the code or binary, runtimes, dependencies, and any other filesystem objects required.</p> <p>Source: https://docker-docs.uclv.cu/get-started/. (Last accessed on May 15, 2025).</p>

U.S. Patent No. 7,721,282 (Claim 1)	
Claim	Example Southwest Count XI Systems and Services
	<p>Docker supports several storage drivers, using a pluggable architecture. The storage driver controls how images and containers are stored and managed on your Docker host. After you have read the storage driver overview, the next step is to choose the best storage driver for your workloads. Use the storage driver with the best overall performance and stability in the most usual scenarios.</p> <p>Source: https://docs.docker.com/storage/storagedriver/select-storage-driver/. (Last accessed on May 15, 2025).</p>
1[b] a first storage unit for storing blocks of a root image of the compute node, wherein the first storage unit comprises a first non-volatile memory, wherein the root image comprises a computer program, wherein the blocks comprise sections of data, and wherein a file of the root image comprises at least one block;	<p>On information and belief, the Southwest Count XI Systems and Services include “a first storage unit for storing blocks of a root image of the compute node, wherein the first storage unit comprises a first non-volatile memory, wherein the root image comprises a computer program, wherein the blocks comprise sections of data, and wherein a file of the root image comprises at least one block.”</p> <p>Docker containers each have their own private filesystem provided by a Docker image. Docker provides storage drivers that control how the images and containers are stored and managed on a Docker host.</p> <p>Docker supports several storage drivers, using a pluggable architecture. The storage driver controls how images and containers are stored and managed on your Docker host. After you have read the storage driver overview, the next step is to choose the best storage driver for your workloads. Use the storage driver with the best overall performance and stability in the most usual scenarios.</p> <p>Source: https://docs.docker.com/storage/storagedriver/select-storage-driver/. (Last accessed on May 15, 2025).</p>

U.S. Patent No. 7,721,282 (Claim 1)

Claim

Example Southwest Count XI Systems and Services



Source: <https://docs.docker.com/guides/docker-overview/>. (Last accessed on May 15, 2025).

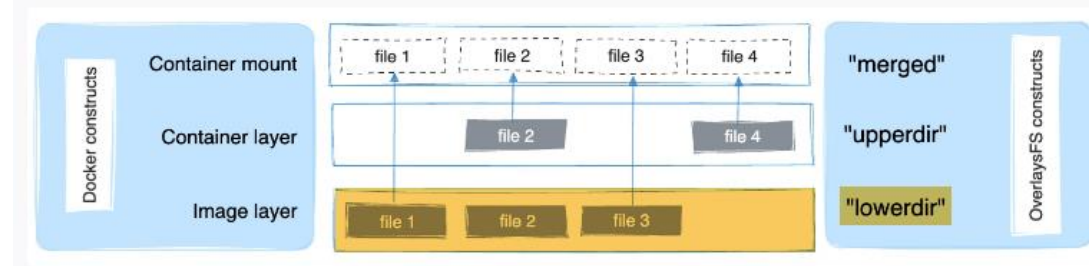
For example, Docker stores images and containers in memory on the host system. Particularly, Docker, through its OverlayFS (Overlay File System), presents all the directories as a single consolidated directory. The lower directory is called ‘lowerdir’ and the upper directory is called ‘upperdir’. The ‘lowerdir’ is the layer that stores blocks of the base or root image. The base or root image includes the application code, dependencies, and other necessary elements for execution or runtime. Each layer, or the files therein, occupies a portion of the ‘lowerdir’.

U.S. Patent No. 7,721,282 (Claim 1)	
Claim	Example Southwest Count XI Systems and Services
	<p>Image and container layers on-disk</p> <p>After downloading a five-layer image using <code>docker pull ubuntu</code>, you can see six directories under <code>/var/lib/docker/overlay2</code>.</p> <div> <p>⊗ Warning</p> <p>Don't directly manipulate any files or directories within <code>/var/lib/docker/</code>. These files and directories are managed by Docker.</p> </div> <pre> \$ ls -l /var/lib/docker/overlay2 total 24 drwx----- 5 root root 4096 Jun 20 07:36 223c2864175491657d238e2664251df13b63adb8d050924f drwx----- 3 root root 4096 Jun 20 07:36 3a36935c9df35472229c57f4a27105a136f5e4dbef0f8796 drwx----- 5 root root 4096 Jun 20 07:36 4e9fa83caff3e8f4cc83693fa407a4a9fac9573deaf48156 drwx----- 5 root root 4096 Jun 20 07:36 e8876a226237217ec61c4baf238a32992291d059fdac95ec drwx----- 5 root root 4096 Jun 20 07:36 eca1e4e1694283e001f200a667bb3cb40853cf2d1b12c29f drwx----- 2 root root 4096 Jun 20 07:36 1 </pre> <p>Source: https://docs.docker.com/storage/storagedriver/overlayfs-driver/. (Last accessed on May 15, 2025).</p>

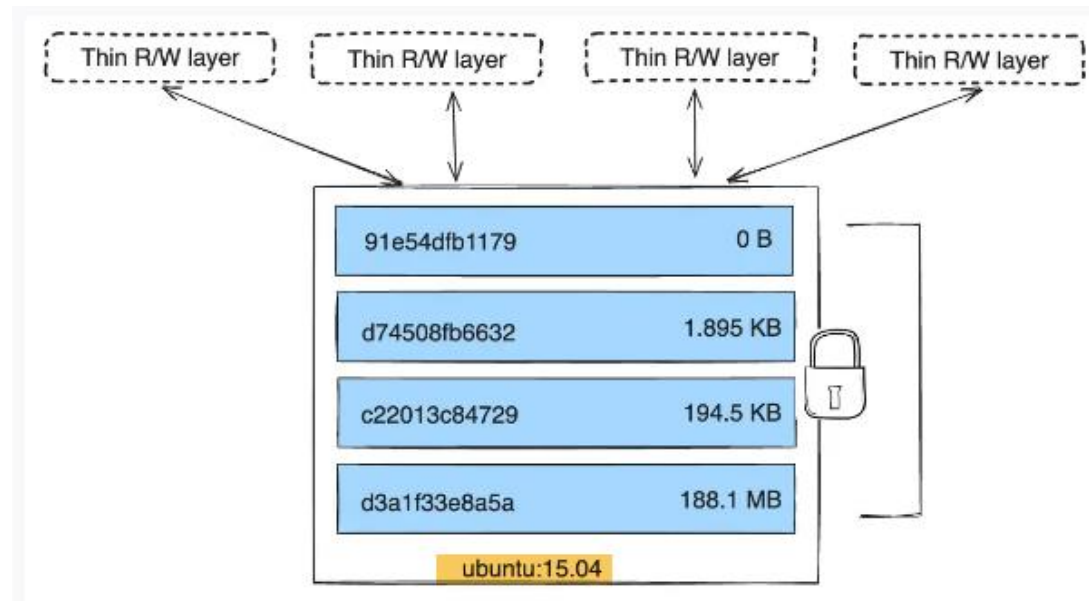
U.S. Patent No. 7,721,282 (Claim 1)

Claim

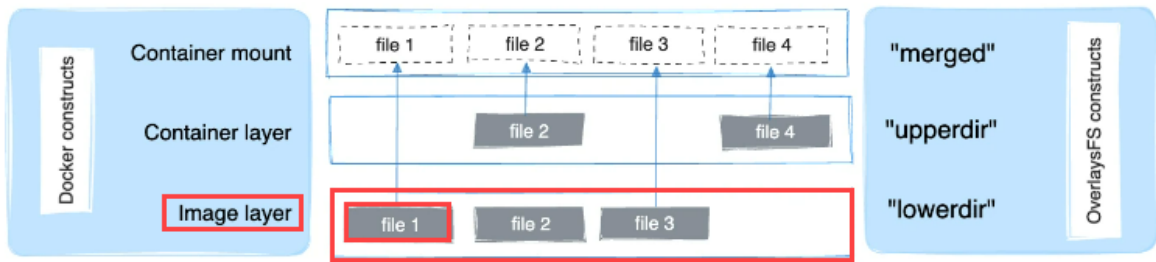
Example Southwest Count XI Systems and Services



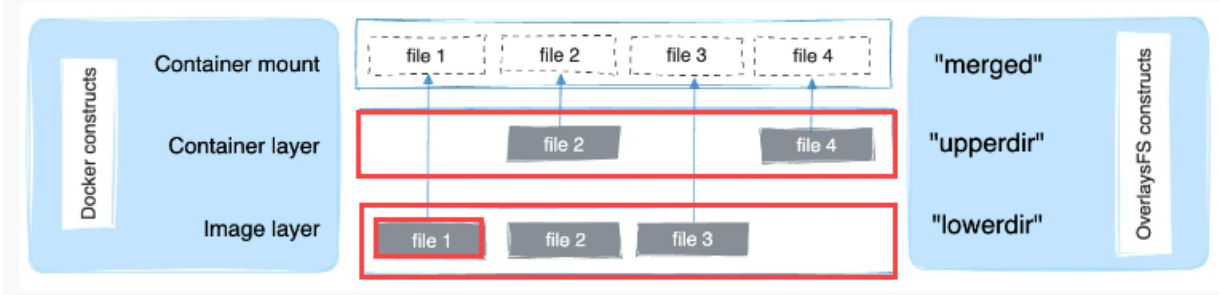
Source: <https://docs.docker.com/storage/storagedriver/overlayfs-driver/>. (Last accessed on May 15, 2025).

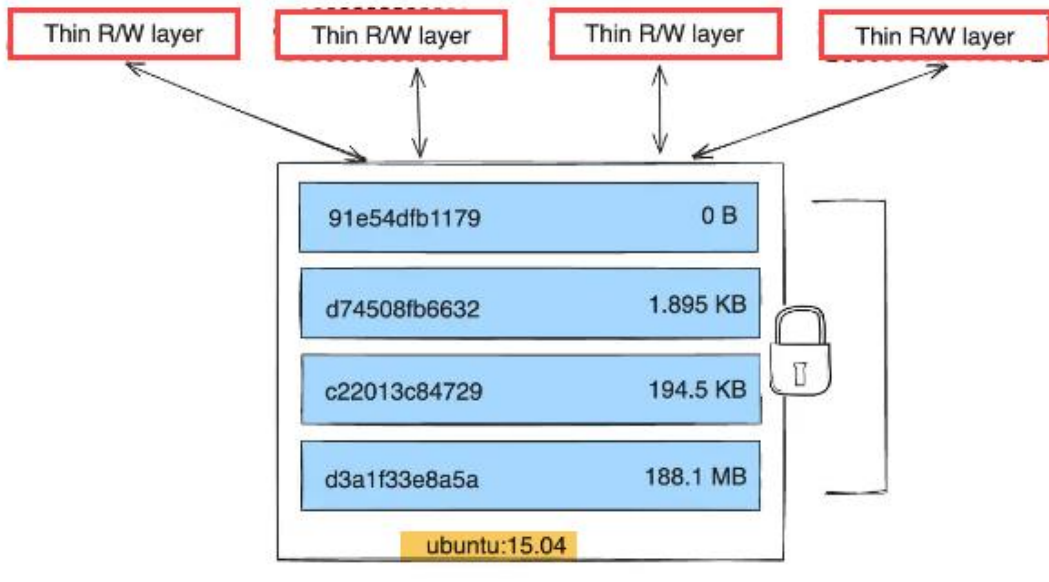


Source: <https://docs.docker.com/storage/storagedriver/>. (Last accessed on May 15, 2025).

U.S. Patent No. 7,721,282 (Claim 1)	
Claim	Example Southwest Count XI Systems and Services
	<p>As shown in the evidence below, each layer can include multiple files. For example, an image layer can include three files, where each of these files includes at least one block of data. A user can see a consolidated view via the merged directory, which is a combined view of the 'lowerdir' and 'upperdir' via overlay2, the storage driver in OverlayFS.</p> <p>The following diagram shows how a Docker image and a Docker container are layered. The image layer is the <code>lowerdir</code> and the container layer is the <code>upperdir</code>. If the image has multiple layers, multiple <code>lowerdir</code> directories are used. The unified view is exposed through a directory called <code>merged</code> which is effectively the containers mount point.</p>  <p>Source: https://docs.docker.com/storage/storagedriver/overlayfs-driver/. (Last accessed on May 15, 2025).</p> <p>Image and container layers on-disk</p> <p>After downloading a five-layer image using <code>docker pull ubuntu</code>, you can see six directories under <code>/var/lib/docker/overlay2</code>.</p> <p>Source: https://docs.docker.com/storage/storagedriver/overlayfs-driver/. (Last accessed on May 15, 2025).</p>

U.S. Patent No. 7,721,282 (Claim 1)	
Claim	Example Southwest Count XI Systems and Services
1[c] a second storage unit for storing a leaf image, the leaf image comprising new data blocks and changes to the blocks of the root image, wherein the second storage unit comprises a second non-volatile memory; and	<p>On information and belief, the Southwest Count XI Systems and Services include “a first storage unit for storing blocks of a root image of the compute node, wherein the first storage unit comprises a first non-volatile memory, wherein the root image comprises a computer program, wherein the blocks comprise sections of data, and wherein a file of the root image comprises at least one block.”</p> <p>Docker includes a top writable layer, where all writes to a container (e.g., a write to add new data or modify existing data) are stored in the writable layer. The writable layer contains only the modified data per the changes made by the respective container instance. The writable layer is stored in memory.</p> <div data-bbox="718 630 1150 680"> <h3>Container and layers</h3> </div> <div data-bbox="718 709 1810 820"> <p>The major difference between a container and an image is the top writable layer. All writes to the container that add new or modify existing data are stored in this writable layer. When the container is deleted, the writable layer is also deleted. The underlying image remains unchanged.</p> </div> <p>Source: https://docs.docker.com/storage/storagedriver/. (Last accessed on May 15, 2025).</p> <div data-bbox="672 915 1848 1120"> <p>Each layer is only a set of differences from the layer before it. Note that both <i>adding</i>, and <i>removing</i> files will result in a new layer. In the example above, the <code>\$HOME/.cache</code> directory is removed, but will still be available in the previous layer and add up to the image's total size. Refer to the Best practices for writing Dockerfiles and use multi-stage builds sections to learn how to optimize your Dockerfiles for efficient images.</p> </div> <div data-bbox="672 1166 1852 1326"> <p>The layers are stacked on top of each other. When you create a new container, you add a new writable layer on top of the underlying layers. This layer is often called the "container layer". All changes made to the running container, such as writing new files, modifying existing files, and deleting files, are written to this thin writable container layer. The diagram below shows a container based on an <code>ubuntu:15.04</code> image.</p> </div> <p>Source: https://docs.docker.com/storage/storagedriver/. (Last accessed on May 15, 2025).</p>

U.S. Patent No. 7,721,282 (Claim 1)	
Claim	Example Southwest Count XI Systems and Services
	<p>For example, the changes, edits, or modifications made to a container will be stored in the ‘upperdir’ of that container. The ‘upperdir’ includes only the changed, edited, or modified file(s) by the container.</p> <p>The following diagram shows how a Docker image and a Docker container are layered. The image layer is the <code>lowerdir</code> and the container layer is the <code>upperdir</code>. If the image has multiple layers, multiple <code>lowerdir</code> directories are used. The unified view is exposed through a directory called <code>merged</code> which is effectively the containers mount point.</p>  <p>Source: https://docs.docker.com/storage/storagedriver/overlayfs-driver/. (Last accessed on May 15, 2025).</p>

U.S. Patent No. 7,721,282 (Claim 1)	
Claim	Example Southwest Count XI Systems and Services
	 <p>The diagram illustrates a Docker storage architecture. At the top, four red-outlined boxes labeled 'Thin R/W layer' are arranged horizontally. Below them is a large white box representing the base image, labeled 'ubuntu:15.04' at the bottom. Inside this box are four blue-outlined boxes representing image layers, each with a hash and a size: '91e54dfb1179' (0 B), 'd74508fb6632' (1.895 KB), 'c22013c84729' (194.5 KB), and 'd3a1f33e8a5a' (188.1 MB). A padlock icon is positioned to the right of the layers, indicating they are locked or read-only. Double-headed arrows connect each 'Thin R/W layer' box to the top of the base image box, showing that each container's writable layer is built on top of the base image.</p> <p>Source: https://docs.docker.com/storage/storagedriver/. (Last accessed on May 15, 2025).</p> <p>For example, below describes how overlay2 functions by comparing and calculating the differences or changes between the parent and its corresponding layer, if it was modified.</p>

U.S. Patent No. 7,721,282 (Claim 1)	
Claim	Example Southwest Count XI Systems and Services
	<pre> // DiffSize calculates the changes between the specified id // and its parent and returns the size in bytes of the changes // relative to its base filesystem directory. func (d *Driver) DiffSize(id, parent string) (int64, error) { if useNaiveDiff(d.home) !d.isParent(id, parent) { return d.naiveDiff.DiffSize(id, parent) } return directory.Size(context.TODO(), d.getDiffPath(id)) } // Diff produces an archive of the changes between the specified // layer and its parent layer which may be "". func (d *Driver) Diff(id, parent string) (io.ReadCloser, error) { if useNaiveDiff(d.home) !d.isParent(id, parent) { return d.naiveDiff.Diff(id, parent) } } // Changes produces a list of changes between the specified layer and its // parent layer. If parent is "", then all changes will be ADD changes. func (d *Driver) Changes(id, parent string) ([]archive.Change, error) { return d.naiveDiff.Changes(id, parent) } </pre> <p>Source: moby/daemon/graphdriver/overlay2/overlay.go at master · moby/moby · GitHub. (Last accessed on May 15, 2025).</p>

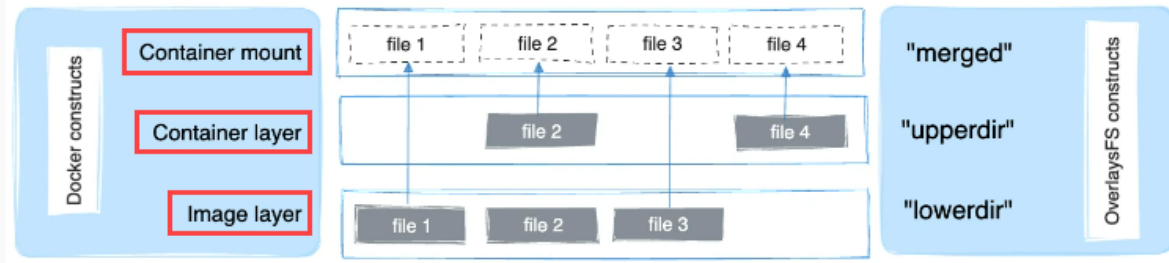
U.S. Patent No. 7,721,282 (Claim 1)	
Claim	Example Southwest Count XI Systems and Services
<p>1[d] a union block device for interfacing between the compute node and the first and second storage units to distribute the application environment to the compute node, wherein the union block device comprises a driver, wherein the union block device creates the application environment by merging the blocks of the root image stored on the first storage unit with the blocks of the leaf image stored on the second storage unit; the union block device comprises a low-level driver for interfacing between the first and second storage units and the file system of the compute node; and the union block device, upon receiving a write request from the compute node for a sector X, creates an appropriate persistent mapping for sector X.</p>	<p>On information and belief, the Southwest Count XI Systems and Services include “a union block device for interfacing between the compute node and the first and second storage units to distribute the application environment to the compute node, wherein the union block device comprises a driver, wherein the union block device creates the application environment by merging the blocks of the root image stored on the first storage unit with the blocks of the leaf image stored on the second storage unit; the union block device comprises a low-level driver for interfacing between the first and second storage units and the file system of the compute node; and the union block device, upon receiving a write request from the compute node for a sector X, creates an appropriate persistent mapping for sector X.”</p> <p>Docker enables building, running, and sharing applications using containers. Docker provides a storage driver – overlay2, for managing and storing images for use. The overlay2 storage driver provides a mechanism for interfacing between the container and storage units.</p> <div data-bbox="659 776 1864 1218"> <h3>How the overlay2 driver works</h3> <p>OverlayFS layers two directories on a single Linux host and presents them as a single directory. These directories are called layers, and the unification process is referred to as a union mount. OverlayFS refers to the lower directory as <code>lowerdir</code> and the upper directory as <code>upperdir</code>. The unified view is exposed through its own directory called <code>merged</code>.</p> <p>The <code>overlay2</code> driver natively supports up to 128 lower OverlayFS layers. This capability provides better performance for layer-related Docker commands such as <code>docker build</code> and <code>docker commit</code>, and consumes fewer inodes on the backing filesystem.</p> </div> <p>Source: https://docs.docker.com/storage/storagedriver/overlayfs-driver/. (Last accessed on May 15, 2025).</p>

U.S. Patent No. 7,721,282 (Claim 1)

Claim

Example Southwest Count XI Systems and Services

The following diagram shows how a Docker image and a Docker container are layered. The image layer is the `lowerdir` and the container layer is the `upperdir`. If the image has multiple layers, multiple `lowerdir` directories are used. The unified view is exposed through a directory called `merged` which is effectively the containers mount point.



Source: <https://docs.docker.com/storage/storagedriver/overlayfs-driver/>. (Last accessed on May 15, 2025).

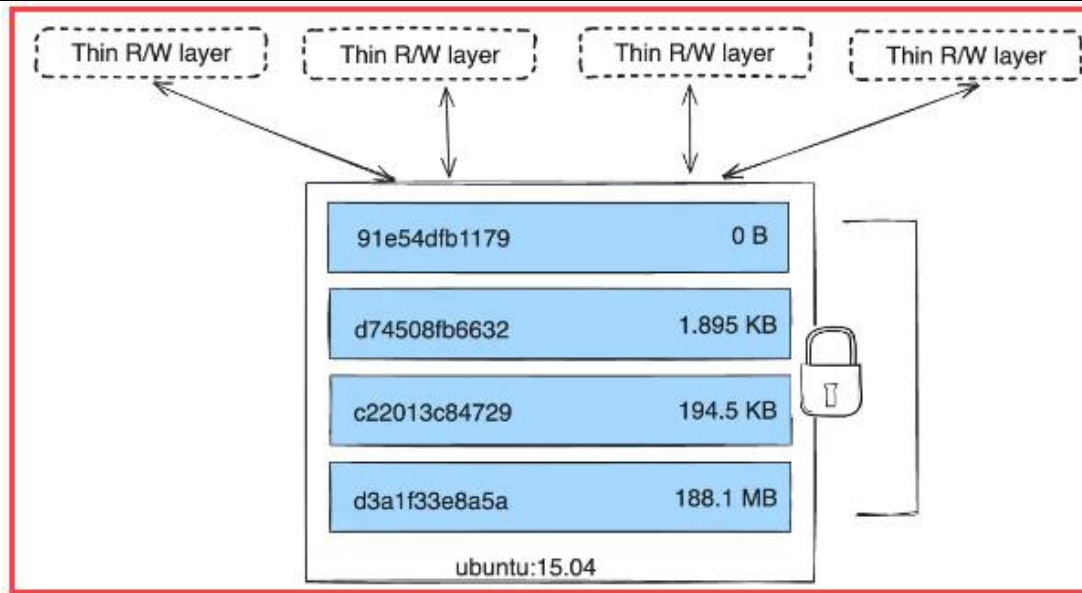
To create a container, the `overlay2` driver combines the directory representing the image's top layer plus a new directory for the container. The image's layers are the `lowerdirs` in the overlay and are read-only. The new directory for the container is the `upperdir` and is writable.

Source: <https://docs.docker.com/storage/storagedriver/overlayfs-driver/>. (Last accessed on May 15, 2025).

U.S. Patent No. 7,721,282 (Claim 1)

Claim

Example Southwest Count XI Systems and Services



Source: <https://docs.docker.com/storage/storagedriver/>. (Last accessed on May 15, 2025).

Images and containers

Fundamentally, a container is nothing but a running process, with some added encapsulation features applied to it in order to keep it isolated from the host and from other containers. One of the most important aspects of container isolation is that each container interacts with its own private filesystem; this filesystem is provided by a Docker **image**. An image includes everything needed to run an application - the code or binary, runtimes, dependencies, and any other filesystem objects required.

Source: <https://docker-docs.uclv.cu/get-started/>. (Last accessed on May 15, 2025).

Docker includes an OverlayFS union filesystem that includes a storage driver, overlay2. OverlayFS includes the image layer and the container layer, which are presented as a single directory in a process

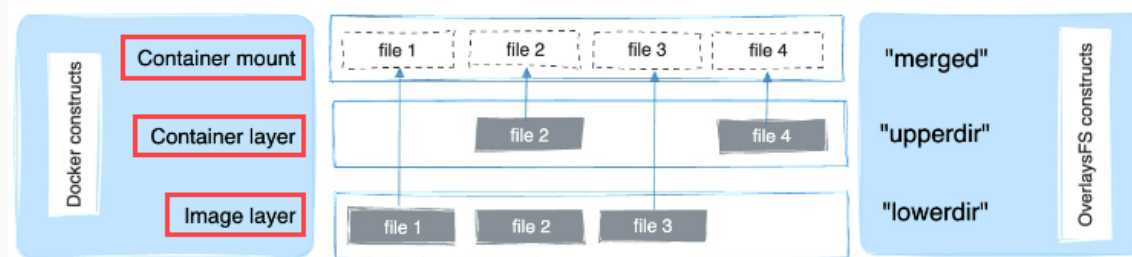
U.S. Patent No. 7,721,282 (Claim 1)	
Claim	Example Southwest Count XI Systems and Services
	<p>called a union mount. In a container filesystem, the overlay2 storage driver combines directories representing the image's read-only layer and writable layers.</p> <div> <h2>Use the OverlayFS storage driver</h2> <p>OverlayFS is a union filesystem.</p> <p>This page refers to the Linux kernel driver as <code>OverlayFS</code> and to the Docker storage driver as <code>overlay2</code>.</p> <h2>How the <code>overlay2</code> driver works</h2> <p>OverlayFS layers two directories on a single Linux host and presents them as a single directory. These directories are called layers, and the unification process is referred to as a union mount. OverlayFS refers to the lower directory as <code>lowerdir</code> and the upper directory as <code>upperdir</code>. The unified view is exposed through its own directory called <code>merged</code>.</p> <p>The <code>overlay2</code> driver natively supports up to 128 lower OverlayFS layers. This capability provides better performance for layer-related Docker commands such as <code>docker build</code> and <code>docker commit</code>, and consumes fewer inodes on the backing filesystem.</p> <p>Source: https://docs.docker.com/storage/storagedriver/overlayfs-driver/. (Last accessed on May 15, 2025).</p> <p>For example, Docker refers to the base image layers as 'lowerdir' and the container layer as 'upperdir,' and the unified view is referred to as merged.</p> </div>

U.S. Patent No. 7,721,282 (Claim 1)

Claim

Example Southwest Count XI Systems and Services

The following diagram shows how a Docker image and a Docker container are layered. The image layer is the `lowerdir` and the container layer is the `upperdir`. If the image has multiple layers, multiple `lowerdir` directories are used. The unified view is exposed through a directory called `merged` which is effectively the containers mount point.



Source: <https://docs.docker.com/storage/storagedriver/overlayfs-driver/>. (Last accessed on May 15, 2025).

Furthermore, all writes are made in the top writable layer (container layer / 'upperdir'). Based on information and belief, it is assumed that writes are made in a specific portion of the writable layer, particularly in the case of modifications made to the existing data stored in an already known portion of the writable layer.

Container and layers

The major difference between a container and an image is the top writable layer. All writes to the container that add new or modify existing data are stored in this writable layer. When the container is deleted, the writable layer is also deleted. The underlying image remains unchanged.

U.S. Patent No. 7,721,282 (Claim 1)	
Claim	Example Southwest Count XI Systems and Services
	<p>Source: https://docs.docker.com/storage/storagedriver/. (Last accessed on May 15, 2025).</p> <p>The second-lowest layer, and each higher layer, contain a file called <code>lower</code>, which denotes its parent, and a directory called <code>diff</code> which contains its contents. It also contains a <code>merged</code> directory, which contains the unified contents of its parent layer and itself, and a <code>work</code> directory which is used internally by OverlayFS.</p> <p>Source: https://docs.docker.com/engine/storage/drivers/overlayfs-driver/. (Last accessed on May 15, 2025).</p> <p>The first time a container writes to an existing file, that file does not exist in the container (<code>upperdir</code>). The <code>overlay2</code> driver performs a <code>copy_up</code> operation to copy the file from the image (<code>lowerdir</code>) to the container (<code>upperdir</code>). The container then writes the changes to the new copy of the file in the container layer.</p> <p>However, OverlayFS works at the file level rather than the block level. This means that all OverlayFS <code>copy_up</code> operations copy the entire file, even if the file is large and only a small part of it's being modified. This can have a noticeable impact on container write performance. However, two things are worth noting:</p> <p>Source: https://docs.docker.com/engine/storage/drivers/overlayfs-driver/. (Last accessed on May 15, 2025).</p> <p>For example, the 'lowerdir' directory can be seen by the <code>getLowerDirs</code> function to return all the files associated with the parent or base image.</p>

U.S. Patent No. 7,721,282 (Claim 1)	
Claim	Example Southwest Count XI Systems and Services
	<pre> 468 func (d *Driver) getLowerDirs(id string) ([]string, error) { 469 var lowersArray []string 470 lowers, err := os.ReadFile(path.Join(d.dir(id), lowerFile)) 471 if err == nil { 472 for _, s := range strings.Split(string(lowers), ":") { 473 lp, err := os.Readlink(path.Join(d.home, s)) 474 if err != nil { 475 return nil, err 476 } 477 lowersArray = append(lowersArray, path.Clean(path.Join(d.home, linkDir, lp))) 478 } 479 } else if !os.IsNotExist(err) { 480 return nil, err 481 } 482 return lowersArray, nil </pre> <p>Source: https://github.com/moby/moby/blob/master/daemon/graphdriver/overlay2/overlay.go. (Last accessed on May 15, 2025).</p> <p>Furthermore, overlay2 first looks at if there have been any modifications at the writable layer. It will then merge the lower and upper directories and return that as the mount path of the container. For example, within a container, by listing out the files.</p> <pre> 508 // Get creates and mounts the required file system for the given id and returns the mount path. 509 func (d *Driver) Get(id, mountLabel string) (_ string, retErr error) { </pre>

U.S. Patent No. 7,721,282 (Claim 1)	
Claim	Example Southwest Count XI Systems and Services
	<pre> 527 mergedDir := path.Join(dir, mergedDirName) 528 if count := d.ctr.Increment(mergedDir); count > 1 { 529 return mergedDir, nil 530 } 531 defer func() { 532 if retErr != nil { 533 if c := d.ctr.Decrement(mergedDir); c <= 0 { 534 if mntErr := unix.Unmount(mergedDir, 0); mntErr != nil { 535 logger.Errorf("error unmounting %v: %v", mergedDir, mntErr) 536 } 537 // Cleanup the created merged directory; see the comment in Put's rmdir 538 if rmErr := unix.Rmdir(mergedDir); rmErr != nil && !os.IsNotExist(rmErr) { 539 logger.Debugf("Failed to remove %s: %v: %v", id, rmErr, err) 540 } 541 } 542 } 543 }() 565 mountData := label.FormatMountLabel(opts, mountLabel) 566 mount := unix.Mount 567 mountTarget := mergedDir </pre> <p>Source: https://github.com/moby/moby/blob/master/daemon/graphdriver/overlay2/overlay.go. (Last accessed on May 15, 2025).</p>